# Improved algorithms and analysis for the laminar matroid secretary problem

David Harris[1], Manish Purohit[2]

April 30, 2014

**Abstract**

In a matroid secretary problem, one is presented with a sequence of objects of various weights in a random order, and must choose irrevocably to accept or reject each item. There is a further constraint that the set of items selected must form an independent set of an associated matroid. Constant-competitive algorithms (algorithms whose expected solution weight is within a constant factor of the optimal) are known for many types of matroid secretary problems. We examine the laminar matroid and show an algorithm achieving provably 0.053 competitive ratio.

## 1 Introduction

In the classical secretary problem, one interviews $n$ secretaries sequentially in random order, each order having equal probability. As soon as one interviews a secretary, one learns the skill level of that secretary, relative to all previously seen applicants. At this point the interviewer must make an irrevocable decision whether or not to hire. The goal is to hire the best secretary.

For this problem, [1],[2],[3] discuss the elegant optimal algorithm. This algorithm looks at the first $\frac{n}{e}$ secretaries, rejects them all, and then from among the remaining secretaries chooses the first one who is better than each of the first observed $\frac{n}{e}$ secretaries (if any). This simple algorithm hires the best secretary with probability $\frac{1}{e}$.

[1]Department of Applied Mathematics, University of Maryland, College Park, MD 20742. Email: `davidgharris29@hotmail.com`.

[2]Department of Computer Science, University of Maryland, College Park, MD 20742. Email: `manishp@cs.umd.edu`.

One of the many generalizations of the secretary problem is called the matroid secretary problem. Here, we are given a matroid $\mathfrak{M}(\mathcal{U}, \mathcal{I})$ (which is known completely beforehand). The ground set also contains weights for each element, which are unknown a priori. The elements arrive one by one in a random order. We denote this ordering by $\pi$, a permutation on $n$ elements. Each element reveals its weight when it arrives. As before we must make an irrevocable decision whether to accept or reject the element when it arrives. The goal is to choose an independent set of the largest weight.

A matroid is a particularly attractive setting for the secretary problem, because of the exchange property. This ensures that even if we make a bad decision about which element to accept, we are not locked in to a bad solution set. In matroid secretary problems, as opposed to more general secretary problems, we can often find a solution set which is relatively close to the optimal one.

The matroid secretary problem can be viewed as a simple model for irrevocable decisions in the presence of uncertainty as to future opportunities. The use of random permutation is conceptually simple, but allows powerful bounds with a minimum of auxiliarly information. Other models which may include prior distributions on the price structure are possible.

For secretary problems, we define the *competitive ratio* to be the ratio of the expected weight obtained by our algorithm, divided by the optimal weight. We note that in the classical secretary problem, one has a $1/e$ chance of choosing the best applicant; for the matroid secretary problem, we do not care about the probability of selecting the largest-weight independent set from the matroid, only in selecting sets which have large weight on average. Furthermore, we do not need any probability of obtaining a large-weight set (other than is implied by Markov's inequality).

For general matroids, [5] gives an $O(\sqrt{\log r})$-competitive algorithm where $r$ is the rank of the matroid. For many special classes of matroids, constant-competitive algorithms are known. In particular, [6] provides a $\frac{3}{16000}$-competitive algorithm for laminar matroids. An alternative algorithm has been demonstrated in [4], which gives a 0.070-competitive algorithm for the laminar matroid.

We improve the algorithm of [6] and obtain a tighter analysis, showing a 0.053-competitive algorithm for the laminar matroid. This improves on [6] by nearly 300-fold. This nearly brings the algorithm of [6] to parity with the new algorithm of [4].

## 2 Definitions and Notation

We let $U$ be the ground set and $w : U \to \mathbb{R}$ be the weight function. Then a laminar matroid is defined by a family $\mathcal{F}$ of laminar subsets. That is, for any $A, B \in \mathcal{F}$ we have $A \subseteq B$ or $B \subseteq A$ or $A \cap B = \emptyset$. In other words, the sets in $\mathcal{F}$ are nested within each other. Each set $A \in \mathcal{F}$ has an associated *capacity* $\mu(A)$. A set $X \subseteq U$ is an independent set in the matroid iff $|X \cap A| \leq \mu(A)$ for all $A \in \mathcal{F}$.

Without loss of generality we may assume $\mu(A) < \mu(B)$ for any $A, B \in \mathcal{F}$ and $A \subseteq B$; for, otherwise $A$ is redundant and may be removed from $\mathcal{F}$.

We use the terminology of [6]. For $i \in U$, we let $M(i)$ denote the minimal set $B \in \mathcal{F}$ such that $i \in B$. We say that $B_1 \in \mathcal{F}$ is a child of $B_2 \in \mathcal{F}$ if $B_1 \subsetneq B_2$ and there exists no intermediate set $B' \in \mathcal{F}$ such that $B_1 \subseteq B' \subseteq B_2$. Naturally $B_2$ is called parent of $B_1$.

For any $A, B \in \mathcal{F}$ such that $A \subseteq B$, we define Chain$[A, B]$ to be the sequence of sets in $\mathcal{F}$ starting with $A$ and ending with $B$ where each set is a child of the following set. In order to denote all sets in $\mathcal{F}$ that contain $i$, we may interchangeably use Chain$[M(i), U]$ or $\mathcal{F}(i)$. To save notation, let OPT denote the optimal solution itself or the total weight of the optimal solution depending on the context. For any $V \subseteq U$ and $B \in F$, let $\mathrm{OPT}_V(B)$ denote the optimal feasible solution that can be obtained from $V \cap B$. For simplicity of notation, let $\mathrm{OPT}(B) = \mathrm{OPT}_U(B)$. Let $\pi$ denote the random ordering of elements in $U$.

## 3 Algorithm

**Algorithm 1:** KickNext Algorithm

```
Let Draw $t \sim \mathrm{Binom}(n, 1-p)$ and let $S = \{\pi(1), \ldots, \pi(t)\}$.
foreach $B \in \mathcal{F}$ do
    let $R(B) \leftarrow OPT_S(B)$
end
foreach $i \in T = U - S$ (taken in the random order $\pi$) do
    foreach $B \in Chain[M(i), U]$ do
        if $R(B) \neq \emptyset$ and $w(i)$ is greater than some element of $R(B)$
        then
            Add $i$ to $SOL(B)$
            Remove the largest element of weight less than $w(i)$ from
            $R(B)$
        else
            break the loop (go to the next item $i$)
        end
    end
end
Return $SOL(U)$;
```

Here, we take the first $1-p$ proportion of items for the sampling phase (used to estimate statistical information about the optimal solution), and we take the latter fraction $p$ to actually build the optimal solution. As we will see, the optimal choice of $p$ is about $p \approx 0.08$. From the sampled set of elements $S$, we calculate $OPT_S(B)$ as the reference set $R(B)$.

We denote the $S = \{\pi(1), \ldots, \pi(t)\}$. Such elements are used for sampling and building statistical information about the optimal set. The remaining items $T = U - S$ are considered for actual selection.

Note that this algorithm does not use the "AddIt" method used in [6], in which during the second phase items enter the optimal solution with some probability less than one. The intuitive explanation for this difference is that any element which is not eligible for the optimal solution should be used to build statistical information, and not simply discarded.

We will briefly explain the intuition behind this algorithm. In the initial sampling phase, we build up a set which looks like the globally optimal solution; in the second phase, we try to mimic the sample optimum as closely as possible. The rule for evicting elements from $R(B)$ appears strange, in that it would be more natural to remove the *lowest-weight* element from

$R(B)$ when inserting a new element. However, if we did this, then for low-weight elements $R(B)$ would become distorted compared to $OPT(B)$. The key innovation of [6] was in using this counter-intuitive eviction rule.

## 4 Analysis

Note that the algorithm selects an element by kicking out a smaller element in $R(B)$ for all $B \in \mathcal{F}(i)$. An element $i$ is not selected to be in $SOL(B)$ iff all elements with weight smaller than $w_i$ in $R(B)$ have been kicked out already.

We assume that, at the end of the sampling phase, we have $|OPT_S(B)| = \mu(B)$ exactly for all $B \in \mathcal{F}$. We can force this to occur with probability one by adding infinitely many elements of infinitesimal weight to the matroid, which will not affect the algorithm's behavior. This simplifying assumption allows us to avoid some corner cases.

Finally, we assume that items have distinct weights; this can be achieved by adding infinitesimal perturbations to the original weights. This affects the behavior of the optimal algorithm only infinitesimally. The perturbation may affect the behavior of this algorithm substantially, as it is based on determining hard cut-off values for whether to accept an element. However, it will suffice to show a good competitive ratio on the perturbed weights.

For a given set $B \in \mathcal{F}$, most elements $x \in T$ will be immediately disqualified from affecting $B$ in any way. We can note a simple condition on element $x \in B$ affecting the set $\mathrm{SOL}(B)$ is that the weight of $x$ exceeds the smallest weight element of $\mathrm{OPT}_S(B')$, for all $B'$ in the chain between $M(x)$ and $B$. We call such elements *qualifying for $B$*. We can bound the number of such qualifying elements as follows:

**Lemma 4.1.** *Consider any set $B \in \mathcal{F}$ and element $i \in U$. Let $OPT_S(B) = \{a_1, a_2, \ldots, a_m\}$ sorted so that $w(a_1) < w(a_2) < \cdots < w(a_m)$. For notational convenience, set $w(a_{m+1}) = \infty$. Let $N_j \subseteq T$, for $j = 1, \ldots, m$, denote the elements $x$ which satisfy the following conditions:*

1. *$x$ qualifies for $B$*
2. *$w(a_j) < x < w(a_{j+1})$*
3. *$x \in T$*

*Then for any non-negative integers $n_1, \ldots, n_m$, we have*

$$P(|N_1| = n_1 \wedge \cdots \wedge |N_m| = n_m \mid i \notin S) \leq p^{n_1 + \cdots + n_m}$$

*Proof.* It suffices to show that, for any $j = 1, \ldots, m$, the probability that $|N_j| = n_j$, conditional on $i \notin S$ as well as $|N_{j+1}| = n_{j+1}, \ldots, |N_m| = n_m$, is at most $p^{n_j}$.

Note that $N_j$ is determined solely by the elements of weight less than $w(a_{j+1})$. Suppose we condition on some choice of $a_{j+1}, \ldots, a_m$. Now $N_j$ depends solely on the positions of elements with weights less than $w(a_{j+1})$, and in particular is independent of $N_{j+1}, \ldots, N_m$. Then $a_j$ is the element of $U$ satisfying the five conditions:

1. $a_j \neq i$
2. $w(a_j) < w(a_{j+1})$
3. $\{a_j, a_{j+1}, \ldots a_m\} \in \mathcal{I}$
4. $a_j \in S$
5. $a_j$ has maximal weight among all that satisfy (1) — (4).

(Condition (1) is redundant, as $i \notin S$ and $a_1, \ldots, a_m \in S$.) We now claim that any qualifying element $x \neq i$ such that $w(x) < w(a_{j+1})$ must satisfy $\{x, a_{j+1}, \ldots, a_m\} \in \mathcal{I}$. For, suppose $x$ violates some $\mu(B') = k$, for $B' \subseteq B$. Then this implies that among $\{a_{j+1}, \ldots, a_m\}$ there are exactly $k$ elements in $B'$. In particular, $x$ does not qualify for $B' \subseteq B$.

Now consider the set $X \subseteq U$ consisting of all elements $x$ which satisfy

$$x \neq i, w(x) < w(a_{j+1}), \{x, a_{j+1}, \ldots, a_m\} \in \mathcal{I}.$$

As we have seen, $a_j$ is the element of $X \cap S$ of largest weight and $n_j$ is the number of elements of $X$ of greater weight than $a_j$.

If $|X| \leq n_j$, then the probability that $|N_j| = n_j$ is zero. Otherwise, we can view this as the following process. Suppose we sort the elements of $X$ in order of decreasing weight. Starting with the largest element of $X$, we assign elements to either $S$ or $T$. These assignments to $S$ are independent with probability $1 - p$. Then $|N_j| = n_j$ iff we assign the first $n_j$ elements to $T$ (probability $p$) and the $(n_j + 1)$th (if it exists) to $S$, which occurs with probability at most $p^{n_j}$.

Hence, conditional on any $a_{j+1}, \ldots, a_m$, the probability that $|N_j| = n_j$ is at most $p^{n_j}$.

$\square$

## 4.1 Probability of selecting an item

Define the *backward rank* of element $i$ for $B \in \mathcal{F}$, denoted as $\text{brank}(i, B)$, to be the number of elements in $\text{OPT}(B)$ having weight less than $w_i$. Similarly, let $\text{brank}_S(i, B)$ be the number of elements in $\text{OPT}_S(B)$ having weight less than $w_i$. It can be easily seen that $\text{brank}_S(i, B) \geq \text{brank}(i, B)$. Furthermore, if $i \in T$, then $\text{brank}_S(i, B) \geq \text{brank}(i, B) + 1$ (proved in [6]). Intuitively, an element $i$ is more likely to be picked by the algorithm if its $\text{brank}(i, B)$ is large.

Now, when element $i \in T$ is considered for inclusion in the solution set, it will be rejected iff there is some $B \in \mathcal{F}(i)$ such that all elements in $OPT_S(B)$ of weight less than $w(i)$ have been evicted already. Let $\textsc{AllKicked}(i, B)$ denote this bad event. We can bound the probability of this event as follows.

**Lemma 4.2.** *Suppose $p < 1/2$. Consider any $B \in \mathcal{F}$ and $i \in OPT$. Now if we define*

$$\alpha = \left(\frac{p + (1-p)\log(1-p)}{2(1-p)p^2}\right)$$
$$c = 4p(1-p)$$

*Then we have*

$$P(\textsc{AllKicked}(i, B)) \leq \frac{\alpha c^{brank(i,B)+1}}{1-c}$$

*Proof.* Fix some $i \in OPT$ and let $\text{brank}(i, B) = d$. All the probabilities we calculate in this proof are conditioned on $i \notin S$; we no longer specify this explicitly to simplify the notation.

Let $\text{OPT}_S(B) = \{a_1, \ldots, a_m\}$ sorted so that $w(a_1) < w(a_2) < \cdots < w(a_m)$. Because of the KickNext rule, the item $i$ will go into $SOL(B)$ unless, for some $l \in \{d+1, \ldots, m\}$, there have been at least $l$ items of weight less than $w(a_{l+1})$ added to $\text{SOL}(B)$ before it.

Now consider an element $i' \neq i$. In order for such an $i'$ to have been added to $\text{SOL}(B)$ before $i$, the following events must have occurred:

1. $i'$ is qualifying for $B$
2. $i'$ comes before $i$ in the ordering $\pi$

We view the suffix of the permutation $\pi$ corresponding to $T$ as generated by the following process. Each element $x \in T$ chooses $\rho(x)$ uniformly at random from the real interval $[0, 1]$. We then form the suffix of $\pi$ by sorting by $\rho$. Suppose we condition on a fixed value of $r = \rho(i)$. Now consider an element $i' \neq i$. In order for such an $i'$ to have been added to $SOL(B)$ before $i$, the following events must have occured:

1. $i'$ is qualifying for $B$
2. $\rho(i') < r$.

Let $Q_l$ denote the number of qualifying items other than $i$ with weight $< w(a_{l+1})$ and let $A_l$ denote the number of such items which also have $\rho(i') < i$. We wish to estimate the probability $A_l \geq l$.

By Lemma 4.1, the random variable $Q_l$ is stochastically dominated by the sum of $l$ independent geometric-$p$ random variables. Given a fixed value for $Q_l$, each such qualifying item $i'$ has a probability $r$ of occuring before $i$. Furthermore, these events are independent (conditional on $r$). Hence the probability $P(A_l \geq l | Q_l = k, i \notin S)$ is at most the probability that a binomial random variable, of $k$ trials and probability $r$, exceeds $l$. In effect, the random variable $A_l$ is formed by conjugating a negative binomial random variable $Q_l$ with a binomial-$r$ distribution. The binomial distribution is a conjugate prior for the negative binomial, hence the distribution of $A_l$ is stochastically dominated by the negative binomial distribution of probability $q = \frac{rp}{1-p+rp}$.

We now wish to estimate the probability that $A_l \geq l$. For a negative binomial random variable $A'_l$, the event $A'_l \geq l$ is equivalent to the situation that we flip a biased coin for $2l-1$ times, where the probability of success is $q$, and the total number of successes is at least than $l$; this is a binomial tail probability. Hence we have

$$P(A_l \geq l \mid \rho(i) = r) \leq P(\text{Binomial}(2l-1, q) \geq l-1)$$

Note that as $p < 1/2$, we have $q < 1/2$ as well. By the Chernoff bound the probability of such a deviation is $\exp(-(2l-1)\text{RelEnt}(\frac{l}{2l-1}||q))$. Here RelEnt is the relative entropy function, given by

$$\text{RelEnt}(x||y) = x\log(x/y) + (1-x)\log(\frac{1-x}{1-y})$$

We can simplify this as

$$\begin{aligned} P(A_l \geq l | \rho(i) = r) &\leq \exp(-(2l-1)\text{RelEnt}(\frac{l}{2l-1}||q)) \\ &= \left(\frac{1-l}{(2l-1)(q-1)}\right)^{1-l} \left(-\frac{l}{q-2lq}\right)^{-l} \\ &\leq \frac{1}{2-2q}(4q(1-q))^l \end{aligned}$$

Integrating over $r \in [0,1]$ gives

$$\begin{aligned}
P(A_l \geq l) &\leq \int_r \frac{dr}{2-2q}(4q(1-q))^l \\
&\leq (4p(1-p))^l \int_r \frac{dr}{2-2q}\frac{4q(1-q)}{4p(1-p)} \\
&\leq (-\frac{p-(p-1)\log(1-p)}{2(p-1)p^2})(4p(1-p))^l \\
&= \alpha c^l
\end{aligned}$$

We use the union-bound for the event $\text{AllKicked}(i, B)$:

$$\begin{aligned}
P(\text{AllKicked}(i, B)) &\leq \sum_{l=d+1}^{\infty} P(A_l \geq l \mid i \notin S) \\
&\leq \sum_{l=d+1}^{\infty} \alpha c^l \\
&\leq \frac{\alpha c^{d+1}}{1-c}
\end{aligned}$$

□

## 4.2 Expected weight of SOL

We cannot take any arbitrary element of the optimal solution and show that it is selected with a good probability by our matroid secretary algorithm. Instead, we use a similar strategy to the uniform matroid, and examine the set of high-scoring elements *collectively*. We show that most of these elements (but not any particular one of them) are selected high probability.

We contrast our approach with that of [6], which adopted a hybrid proof strategy between fully analyzing the collective behavior of the optimal solution, and analyzing individual elements of the solution. In [6], certain elements in the optimal solution were identified, referred to as "good" elements, which were shown to have a high probability of being selected by the secretary algorithm. This type of analysis is inherently not tight. We will instead determine the worst possible arrangement of the optimal solution, and show that it still is selected with high probability.

We use our upper bound on the probability of the event AllKicked to obtain a lower bound on the expected weight of our solution:

$$E[w(SOL)] \geq \sum_{i \in \text{OPT}} w(i)[\text{Probability that } i \in \text{SOL}]$$

$$\begin{aligned}
&\geq \sum_{i\in \text{OPT}} w(i) \times p \times [\text{Probability that } i \in \text{SOL} | i \notin S] \\
&\geq \sum_{i\in \text{OPT}} w(i) \times p \times [1 - \sum_{B\in\mathcal{F}(i)} P(\textsc{AllKicked}(i,B))] \\
&\geq p \left[ \sum_{i\in \text{OPT}} w(i) - \sum_{i\in \text{OPT}} \sum_{B\in\mathcal{F}(i)} w(i). \frac{\alpha c^{1+\text{brank}(i,B)}}{1-c} \right] \\
&\geq p \left[ w(\text{OPT}) - \frac{\alpha}{1-c} \sum_{i\in \text{OPT}} w(i) \sum_{B\in\mathcal{F}(i)} c^{1+\text{brank}(i,B)} \right]
\end{aligned}$$

In order to use this estimate, we need to obtain an upper bound on the sum

$$\sum_{i\in \text{OPT}} w(i) \sum_{B\in\mathcal{F}(i)} c^{1+\text{brank}(i,B)}$$

The presence of the weight $w(i)$ complicates things, so as a preliminary we consider the unweighted version of this sum.

Let $\text{OPT}^m_{\text{large}}(B)$ denote the $m$ largest elements in $\text{OPT}(B)$.

**Lemma 4.3.** *Let $B \in \mathcal{F}$ and let $m \geq 0$ be an integer. Define $g(m, B)$ by*

$$g(m,B) = \sum_{i\in OPT^m_{large}(B)} \sum_{B'\in Chain[M(i),B]} c^{1+brank(i,B')}$$

*Suppose $c < 1/2$. Then*

$$g(m,B) \leq \frac{2c}{1-c} |OPT^m_{large}(B)|.$$

*Proof.* For each integer $i$ define $c_i = c + c^2 + \cdots + c^i$, and define $c_\infty = \dfrac{c}{1-c}$.

We will need to show a stronger bound, specifically that for all $B \in \mathcal{F}$ and all $m \geq 0$ we have

$$g(m,B) \leq 2c_1 + \cdots + 2c_{m-1} + c_m + c_m c_{k-m}$$

where $k = \mu(B) \geq m$.

We will show this by induction on the capacity $k$. Note that for a given value of $k$, we are proving the inductive hypothesis simultaneously for all $B \in \mathcal{F}$ and all possible values of $m$.

We view the laminar family as consisting of levels, corresponding to each possible value for the capacity. When computing $g(m, B)$, we have the contribution at level $k$ itself, as well as the contribution from the lower levels. Let $B_1, \ldots, B_j$ be a coarsest $\mathcal{F}$-partition of $B$ (other than $B$ itself). Let $X = \mathrm{OPT}^m_{\mathrm{large}}(B)$, and let $m_i = |B_i \cap X|$ and $k_i = \mu(B_i)$ for each $i = 1, \ldots, j$. For each $i$ we have $X \cap B_i = \mathrm{OPT}^{m_i}_{\mathrm{large}}(B_i)$. By the capacity constraints we must have $m_i \le k_i < k$ for each $i$.

By laminarity we have

$$g(m, B) = \sum_{i \in X} c^{1+\mathrm{brank}(i,B)} + g(m_1, B_1) + \cdots + g(m_j, B_j)$$

The elements of $X$ have maximal bottom-rank in $X$. Hence the term $\sum_{i \in X \cap B} c^{1+\mathrm{brank}(i,B)} = c^k + \ldots c^{k-m+1} = c_m c^{k-m}$. Each $B_i$ has rank less than $k$ so we apply the inductive hypothesis and obtain

$$g(m, B) \le c_m c^{k-m} + \sum_{i=1}^{j} 2c_1 + \cdots + 2c_{m_i-1} + c_{m_i} + c_{m_i} c_{k_i - m_i}$$

The right-hand side is a convex function $m_1, \ldots, m_j$, hence it attains its maximum when these are set to their most extreme possible values. When $m < k$ strictly, we may set $j = 1, m_1 = m, k_1 = k - 1$; when $m = k$, we may set $j = 2, k_1 = k_2 = k - 1, m_1 = m - 1, m_2 = 1$. In the first case, we obtain

$$\begin{aligned} g(m, B) &\le c_m c^{k-m} + \sum_i 2c_1 + \cdots + 2c_{m_i-1} + c_{m_i} + c_{m_i} c_{(k-1)-m_i} \\ &\le c_m c^{k-m} + 2c_1 + \cdots + 2c_{m-1} + c_m + c_m c_{k-1-m} \\ &= 2c_1 + \cdots + 2c_{m-1} + c_m + c_m(c_{k-1-m} + c^{k-m}) \\ &= 2c_1 + \cdots + 2c_{m-1} + c_m + c_m c_{k-m} \end{aligned}$$

In the second case, we obtain

$$\begin{aligned} g(m, B) &\le c_m c^{k-m} + 2c_1 + \cdots + 2c_{m-2} + c_{m-1} + c_{m-1} c_{(k-1)-(m-1)} + c_1 + c_1 c_{(k-1)-1} \\ &= c_m + 2c_1 + \cdots + 2c_{m-2} + c_{m-1} + c_1 + c_1 c_{m-2} \\ &= 2c_1 + \cdots + 2c_{m-2} + c_{m-1} + c_m + c + c c_{m-2} \\ &= 2c_1 + \cdots + 2c_{m-1} + c_m \\ &= 2c_1 + \cdots + 2c_{m-1} + c_m c_{k-m} \end{aligned}$$

as claimed. □

Next we use this unweighted bound to bound the weighted sum:

**Lemma 4.4.** *If $c < 1/2$ we have*

$$\sum_{i \in OPT} w(i) \sum_{B \in \mathcal{F}(i)} c^{1+brank(i,B)} \le \frac{2c}{1-c} w(OPT)$$

*Proof.* Sort the elements of OPT by weight so that $w(x_1) > w(x_2) > \cdots > w(x_l)$. Define $c_\infty = \frac{c}{1-c}$ as above. Then we have

$$\begin{aligned}
&\sum_{i \in \text{OPT}} w(i) \sum_{B \in \mathcal{F}(i)} c^{1+\text{brank}(i,B)} \\
&\quad = w(x_l)g(l,U) + (w(x_{l-1}) - w(x_l))g(l-1,U) + \cdots + (w(x_2) - w(x_1))g(1,U) \\
&\quad \le w(x_l)2lc_\infty + (w(x_{l-1}) - w(x_l))2(l-1)c_\infty + \cdots + (w(x_2) - w(x_1))2c_\infty \\
&\quad = 2c_\infty(w(x_l) + w(x_{l-1}) + \ldots w(x_1)) \\
&\quad = 2c_\infty w(OPT)
\end{aligned}$$

□

We consider the contributions to SOL of the elements of OPT.

**Theorem 4.1.** *The expected value of the weight of SOL is at least a factor $p(1 - \frac{2\alpha c}{(1-c)^2})$ of optimal.*

*Proof.*

$$\begin{aligned}
E[w(SOL)] &\ge \sum_{i \in \text{OPT}} w(i) p(1 - \sum_{B \in \mathcal{F}} P(\text{AllKicked}(i,B))) \\
&\ge p(\sum_{i \in \text{OPT}} w(i) - \sum_{B \in \mathcal{F}} \sum_{i \in \text{OPT} \cap B} w(i) \frac{\alpha}{1-c} c^{\text{brank}(i,B)+1}) \\
&\ge p(w(\text{OPT}) - \frac{\alpha}{1-c} 2c_\infty w(\text{OPT})) \\
&= w(\text{OPT}) p(1 - \frac{2\alpha c}{(1-c)^2})
\end{aligned}$$

□

**Theorem 4.2.** *The KickNext algorithm achieves a competitive ratio of* 0.053

*Proof.* Set $p = 0.08$ and apply Theorem 4.1. □

## References

[1] Dynkin, E. “Optimal choice of the stopping moment of a Markov process” Dokl. Akad. Nauk SSSR 150, pp. 238-240 (1963).

[2] Freeman, P. “The secretary problem and its extensions: a review.” Internat. Statist. Rev. 51(2), pp. 189-206 (1983).

[3] Gardner, M. Mathematical games column. Scientific American Feb., Mar., 35, 1960.

[4] Jaillet, P., Soto, J., Zenklusen, R.: “Advances on Matroid Secretary Problems: Free Order Model and Laminar Case.” arXiv:1207.1333 (2012).

[5] Chakraborty, S., Lachish, O. “Improved competitive ratio for the matroid secretary problem” Symposium on Discrete Algorithms 2012, pp. 1702-1712 (2012)

[6] Im, S. and Wang, Y. “Secretary Problems: Laminar Matroid and Interval Scheduling.” Symposium on Discrete Algorithms 2011, pp. 1265-1274 (2011).

[7] Kleinberg, R. “A multiple-choice secretary algorithm with applications to online auctions” Symposium on Discrete Algorithms 2005, pp. 630-631 (2005).